

NAC10CICD

Laboratório com teste de aplicação node utilizando a integração Git + Travis + Heroku

Neste laboratório executaremos a construção do exercício anterior: Uma aplicação node utilizando repositório Git utilizará o Travis como CI para execução de um teste funcional, este teste será utilizado como "trigger" para o delivery da aplicação utilizando a plataforma PAAS Heroku.

Exercício baseado neste Laboratório: <https://pt.surveymonkey.com/r/KKQH899>;

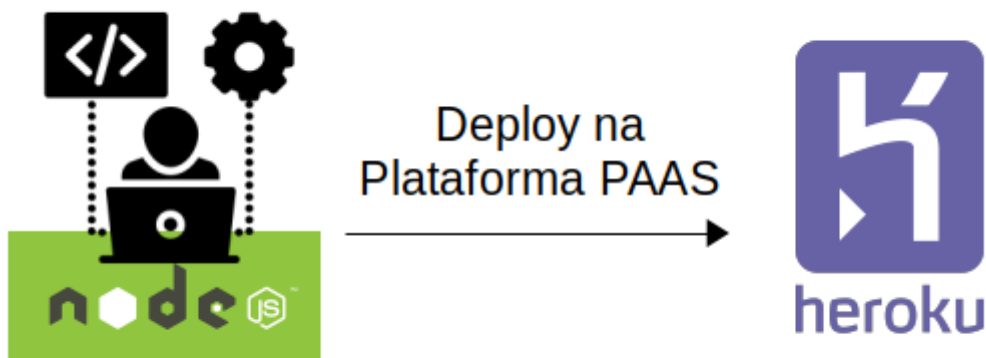


⁝ Pré-requisitos

Para a execução deste laboratório os seguinte requisitos devem ser verificados;

1. Acesso na plataforma [github](#);
2. Acesso na plataforma [heroku](#);
3. Acesso na plataforma [travis](#) (Para este acesso é possível executar login utilizando a conta no Github);
4. Instalação do [Heroku Cli](#)
5. Instalação do [Travis Cli](#);

⁝ Etapa 1 - Entrega da aplicação utilizando o Heroku Cli



Objetivo: Testar a aplicação usando o Heroku (será necessário criar a app antes na plataforma).

1. Faça login no github e execute um fork [deste repositório](#) ele possui a base de código necessária para testar o funcionamento da aplicação e sua entrega de forma direta ou utilizando encapsulamento em containers (repositórios utilizados em aulas anteriores podem ser removidos);
2. Após o Fork clone o repositório localmente no mesmo host onde o Heroku Cli está instalado;
3. A partir do diretório do projeto faça login no Heroku e utilize a sequência abaixo para criar uma app:

```
heroku login -i  
heroku create
```

```
Creating app... done, ● blooming-harbor-25561  
https://blooming-harbor-25561.herokuapp.com/ | https://git.heroku.com/blooming-harbor-25561.git
```

Após o login o segundo comando criará uma aplicação com um nome randomico, isso é necessário pois o nome de um projeto no Heroku app é sempre único;

4. Entregando uma versão preliminar da aplicação de forma direta (sem utilizar o CI):

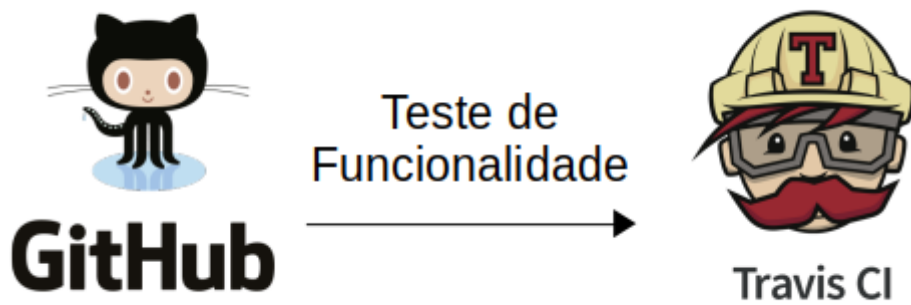
```
git push heroku master
```

Neste processo o Heroku receberá um push com o código base da aplicação, como não houve uma declaração direta da linguagem de programação a plataforma tentará inferir a linguagem com base nos commits executados no repositório master e o processo de deploy deverá ocorrer como esperado;

No caso do node por exemplo, essa inferência da linguagem de programação em uso ocorre devido a estrutura do package.json com pacotes NPM, se você enviasse um aplicativo escrito em PHP com um composer.json essa app seria provavelmente identificada como uma app escrita em PHP. Segundo na documentação oficial da plataforma disponível no endereço <https://devcenter.heroku.com/articles/buildpacks> essa identificação ocorre no primeiro envio devido a ordem na qual os buildpacks são chamados para detecção.

Após o deploy teste a aplicação acessando a URL da aplicação;

↳ Etapa 2 - Integração entre a App e o Travis CI



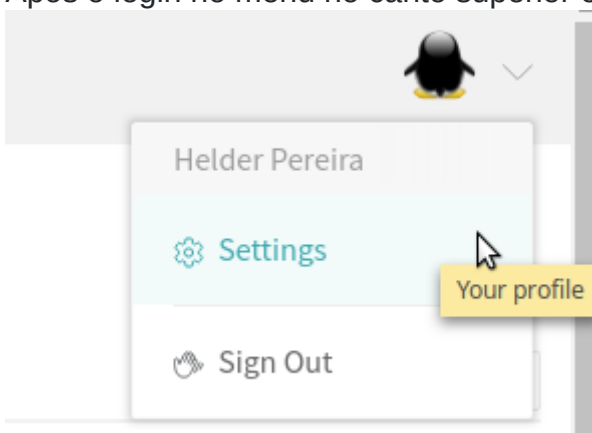
Objetivo: Testar a integração entre o repositório de código e o travis

Para configurar o processo de integração o Travis utiliza como base uma linguagem declarativa, neste caso é criado um arquivo `.travis.yml` na raiz do projeto no Git, essa estrutura pode ser conferida com base na [documentação do CI para javascript/nodejs](#);

O arquivo necessário para esta integração já existe e possui o formato abaixo:


```
language: node_js
node_js:
  - "10"
deploy:
  provider: heroku
  api_key:
    secure: 'KEY'
on:
  all_branches: true
```







1. Faça um acesso na página do Travis pelo endereço: <https://travis-ci.org/>, utilize suas credenciais do Github na opção "Sign in with Github";
2. Após o login no menu no canto superior esquerdo da tela acese a opção Settings:




3. O modelo de integração que usaremos será através de webhooks "Legacy Services Integration", localize o repositório referente ao seu Fork do projeto e habilite a integração usando o "switch button no canto direito da tela":

Legacy Services Integration

 Filter repositories

 classroom	<input type="checkbox"/>	 Settings
 devops-essentials	<input type="checkbox"/>	 Settings
 NAC10CICD	<input type="checkbox"/>	 Settings

- Volte a opção dashboard e procure pelo projeto NAC10CICD, utilize a opção "Trigger a build" para validar a comunicação entre o CI e o repositório **Nesta etapa o processo de deploy da aplicação deverá falhar pois o CI ainda não possui as credenciais de acesso a app no Heroku.

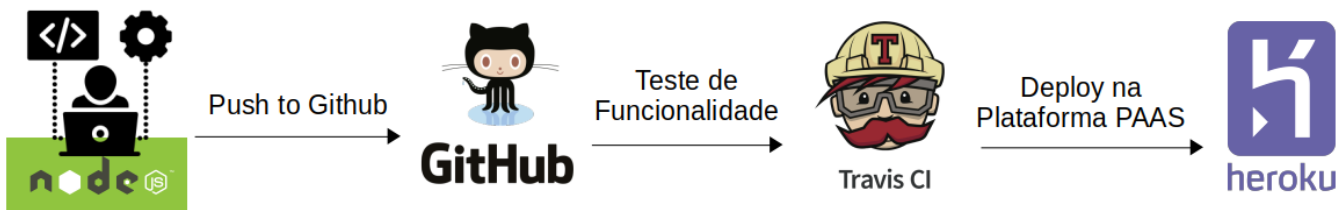
 fiapsecdevops
NAC10CICD

DEFAULT BRANCH
☐ master

There are no builds for this repo yet

Trigger a build

Etapa 3 - Finalizando o processo de Integração



O arquivo de script build `.travis.yml` é responsável pela integração ente o CI e o repositório, neste ponto a ultima etapa do lab é vincular o "aceite de alterações executadas no CI ao processo de deploy no Heroku"

- Para configurar esta etapa da integração será necessário utilizar o cliente do travis para encryptar a token de acesso ao Heroku, no terminal onde os clientes foram instalados faça login no heroku e em seguida execute:

```
heroku login -i
heroku auth:token
```

O comando a seguir executará o mesmo procedimento acima na instrução entre parênteses para em seguida gravar a informação do token obtido na configuração do travis arquivo `".travis.yml"`:

```
travis encrypt $(heroku auth:token) --add deploy.api_key
```

Esse formato está documentado da página do travis referente a deploy usando Heroku: <https://docs.travis-ci.com/user/deployment/heroku/>;

Ao final do processo verifique novamente o arquivo `.travis.yml`, ele possuirá a nova chave criptografada;

2. Para finalizar o processo altere o nome da App no arquivo `.travis.yml`, localize o campo `app: 'NAME_HEROKU_APP'` e modifique o valor de acordo com o nome da aplicação criada no Heroku;
3. **faça o commit das duas auterações executadas** elas serão necessárias para que o ci consiga autenticar e executar o deploy da aplicação;
4. Após a instalação o ultimo processo refere-se a configuração da app para verificar automaticamente o repositório git:

A partir do painel de controle do Heroku na opção Dashboard identifique a App que criamos, navegue até o menu "Deployment method" e escolha a opção "Github":

The screenshot shows the Heroku dashboard for the application 'blooming-harbor-25561'. The 'Deploy' tab is selected. Under 'Add this app to a pipeline', there is a section for 'Add this app to a stage in a pipeline to enable additional features'. Below this, there are three deployment method options: 'Heroku Git' (Use Heroku CLI), 'GitHub' (Connect to GitHub), and 'Container Registry' (Use Heroku CLI). The 'GitHub' option is highlighted with a mouse cursor.

5. No campo "Connect to GitHub" localize o seu repositório para estabelecer a conexão;
6. Marque a opção "Wait for CI to pass before deploy" e habilite o deploy automático utilizando a opção "Enable Automatic Deploys";

The screenshot shows the 'Automatic deploys' configuration page. It explains that a chosen branch will be automatically deployed. A dropdown menu shows 'master' as the selected branch to deploy. The checkbox 'Wait for CI to pass before deploy' is checked. Below this, there is a button labeled 'Enable Automatic Deploys'.

7. Finalizando o processo execute um novo commit na branch master, de preferência alterando o texto entregue quivo index.js e acompanhe em seguida o processo de validação da alteração pelo travis e o deploy da nova versão no Heroku.